REMARKS/ARGUMENTS

A Final Rejection was entered on October 6, 2004 and applicants' attorney responded to the Final Rejection on November 15, 2004. An Advisory Action was sent on January 26, 2005. The Advisory Action propounded that the applicants' arguments did not overcome the rejections in the October 6, 2004 Final Rejection.

In the discretion of the Examiner, applicants' attorney is open to another interview to further highlight the distinctions set forth herein and hopefully expedite allowance of the claims. As indicated, claims 1-26 remain in this application for further review. Claims 1, 13 and 20 have been amended as more fully set forth above.

I. Rejection of Claims 1-26 under 35 U.S.C. 102(b) and 35 U.S.C. 103(a).

Per the Final Rejection dated November 15, 2004, claims 1-3 and 11-13, 15-16 and 20-26 were rejected under 35 U.S.C. 102(b) as being anticipated by Larus, "Whole Program Paths", ACM Sigplan Notices, Vol. 34, No. 5, Atlanta, GA, May 1999, pp. 259-269 ("Larus"). Claims 4-9, 14 and 17-19 were rejected under 35 U.S.C. 103(a) as being unpatentable over Larus in view of U.S. Patent No. 6,247,020 issued to Minard ("Minard"). Applicants respectfully disagree with these rejections.

A. Claim Elements not Taught in Larus

Even though applicants believe that the claims are allowable as written, claims 1, 13 and 20 have been amended in order to clarify the invention and expedite this matter. Applicants' independent claim 1 specifically recites as follows:

> "a stream flow detector that is configured to generate a stream flow output that displays the ***frequency that repetitively occurring data access sequences follow***

other repetitively occurring data access sequences when non-repetitively data access sequences are ignored." (Emphasis added).

Applicants' independent claim 13 specifically recites as follows:

"a stream flow detector that is configured to generate a stream flow output that displays the ***frequency that repetitively occurring data access sequences follow other repetitively occurring data access sequences when non-repetitively data access sequences are ignored.***" (Emphasis added).

Applicants' independent claim 20 specifically recites as follows:

"generating a stream flow output that displays the ***frequency that repetitively occurring data access sequences follow other repetitively occurring data access sequences when non-repetitively data access sequences are ignored.***" (Emphasis added).

The above amendments to the claims do not include new matter. The specification of the present invention specifically recites as follows:

> "Hot data streams 330 may be sent to stream flow graph detector 320 and/or hot data streams abstractor 315. Stream flow graph detector 320 may use the WPS created by path extractor 305 in conjunction with hot data streams 330 to create stream flow graph 325. A stream flow graph shows the number of times in a trace each hot data stream immediately follows another hot data stream, when intervening cold references are ignored. For example, in stream flow graph 325, the hot data stream designated by B' follows the hot data stream designated by A' 4 times and follows itself once. In addition, the hot data stream represented by A' follows the hot data stream represented by B' 5 times.
>
> The following example illustrates this in a relatively simple WPS. Assume a WPS of **CB ABC** EF **CB ABC** FF **CB ABC CB ABC CB** D **CB ABC** (where spacing is added for readability and hot data streams are shown in bold). ABC directly follows CB 5 times and CB directly follows ABC 4 times and itself once (ignoring intervening cold references). In some senses, stream flow graphs may be thought of as control flow graphs for data accesses. In a more complicated stream flow graphs, many hot data streams may be interconnected by edges showing how often each hot data stream follows another." (Page 11, lines 9-25).

The portion of the specification cited above is but one example from the specification and does not represent the breath of any claim term. This citation is for explanatory purposes only

and not meant to impute any limitations into the claims apart from the claim language itself insofar as application asserts that the terms of the claims are clear.

B. The Teachings of Larus

1. Section 3.1 - SEQUITUR Algorithm

Section 3.1 of Larus, does not teach a stream flow output graph that displays "the frequency that repetitively occurring data access sequences follow other repetitively occurring data access sequences when non-repetitively data access sequences are ignored." SEQUITUR is a formula used to represent a data access sequence. SEQUITUR uses formulas provide representations of the sequence or string when elements are repeated. However, the entire string or sequence is still associated with the formula. Stated another way, the entire data access sequence is "unwound" when the formulas are calculated. Elements of the data access sequence are not ignored.

The SEQUITUR algorithm follows two rules. The first rule is referred to as the Digram Uniqueness Property Rule. A digram is a pair of consecutive symbols. For example, in the sequence (abca), the symbols (ab) may be considered a digram. If two of the same digrams exist, SEQUITUR replaces both occurrences of the digram with a non-terminal symbol. For example, digram uniqueness property rule may work as follows:

S = abca	SEQUITUR does not apply because there is not two or more digrams
S = abcab S = AcA A = ab	SEQUITUR applies because two (ab)s exist

In the above example, the symbols (ab) occurs twice. SEQUITUR replaces the symbols (ab) with the symbol (A). The symbol (A) is then set equal to (ab). In this manner, SEQUITUR

can reduce the size of a string through a formula representation when repeating elements exist. However, when the formula is calculated, the entirety of the string still exists. SEQUITUR does not remove any data access sequences; it merely represents the data access sequences in a formula. For example, in the above, symbols (ab) are not removed. They still exist in the formula (A=ab). The sequence is merely represented in a different manner through SEQUITUR.

The second rule of SEQUITUR is referred to as the Utility Property Rule. The Utility Property Rule states that all non-terminal symbols (capitalized letters) in a grammar must be referenced more than once by other rules (i.e. A=ab, B=Ac, C=Ad, or D=BC). SEQUITUR exchanges a rule referenced only once with the rules right side. For example, before the Utility Property Rule is applied, a formula may be as follows:

S = abcabdabcabd
S = DD
A = ab
B = Ac
C = Ad
D = BC

When the rule D=BC is expanded, the string ultimately becomes S=abcabdabcabd. However, when SEQUITUR applies the Utility Property Rule to the above algorithm, the algorithm is as follows:

S = abcabdabcabd
S = DD
A = ab
D = AcAd

The formula represents the same string. When D=AcAd is expanded, the string ultimately becomes S=abcabdabcabd. As is shown in the above example, non-terminal symbols (B) and (C) are no longer used in the formula. Their corresponding symbols (Ac) and (Ad) are

then moved down into symbol (D). In accordance with the utility property rule, non-terminals (B) and (C) were only used once in the formula; therefore, SEQUITUR exchanges these symbols by including their equation in symbol (D). Such a procedure makes the algorithm more efficient for expanding the sequence. However, the above does not indicate that non-repetitively occurring data access sequences are ignored. Both the examples above indicate that S = abcabdabcabd. Elements of the sequence are not ignored. The utility property rule merely indicates that the equation for "unwinding" the sequence is written in a certain manner. The data access sequence, however, is the same regardless of the formula used to represent it.

2. **FIGURE 2 and Whole Program Paths**

Neither FIGURE 2 of Larus nor the discussion pertaining to Whole Program Paths teach or otherwise suggest a stream flow output graph that displays "the frequency that repetitively occurring data access sequences follow other repetitively occurring data access sequences when non-repetitively data access sequences are ignored."

FIGURE 2 depicts a SEQUITUR grammar and a whole program path that represents the SEQUITUR grammar. The Whole Program Path represents the SEQUITUR grammar through arrows. For example, referring to FIGURE 2, the SEQUITUR grammar C = BB is represented in the Whole Program Path by a (C) node having two arrows pointing to the (B) node. In this manner, the entire SEQUITUR sequence may be represented through Whole Program Paths. There is no teaching in FIGURE 2 of a stream flow output graph that displays "the frequency that repetitively occurring data access sequences follow other repetitively occurring data access sequences when non-repetitively data access sequences are ignored."

In conjunction with Figure 5, Larus teaches that "a minimal hot subpath is the shortest prefix of a subpath with cost of C or more." (Larus, at 265). Larus further teaches "minimal hot subpaths are of interest, since longer hot subpaths are easily found by adding acyclic paths to a minimal subpath." (Larus, at 265). In relation to Figure 5, Larus gives an example as follows: Suppose that each acylic path a, b, and c has a cost of 1 and that we are looking for hot subpaths of length greater than 1 and less than 4 whose cost is 6 or more. The WPP contains four overlapping hot subpaths: ab, bc, bb, and ca. The algorithm in this paper identifies two hot subpaths (ab and bc). The other two can be found by extending these two. (Larus, at 264).

With reference to Figure 5, Larus is teaching that the string **abbcabbcabbc** includes four overlapping hot subpaths with a length greater than 1 and less than 4 whose cost is 6 or more (**ab, bc, bb,** and **ca**). Larus then states "this paper identifies two hot subpaths (ab and bc)." and that "[t]he other two can be found by extending these two." (Larus, at 264). Here, applicants assert that Larus is merely giving a succinct example and acknowledging that there is no need to reiterate the example for every possible hot subpath. There is no teaching whatsoever of a stream flow output graph that displays "the frequency that repetitively occurring data access sequences follow other repetitively occurring data access sequences when non-repetitively data access sequences are ignored."

3. **Section 3.2 - SEQUITUR Enhancement.**

Section 3.2 of Larus does not teach a stream flow output graph that displays "the frequency that repetitively occurring data access sequences follow other repetitively occurring data access sequences when non-repetitively data access sequences are ignored." Section 3.2 of Larus pertains to enhancing the SEQUITUR algorithm by looking ahead a single symbol before

introducing a new digram rule. When the algorithm does not look ahead one symbol, representation of the same occurrences can vary, because rules introduced while processing a first occurrence may change the sequences of reductions applied to a second occurrence.

C. **Claims 1-26 are Allowable Over the Cited References**

Independent claims 1, 13 and 20 are clearly allowable over the cited references for the reasons set forth above. Regarding dependent claims 2-3, 11-12, 15-16 and 21-26, claims 2-3, 11-12, 15-16 and 21-26 are not taught or otherwise suggested by Larus. Moreover, claims 2-3, 11-12, 15-16 and 21-26 ultimately depend from independent claims 1, 13 and 20, respectively. Claims 1, 13 and 20 are allowable as previously stated and as such, applicants assert that claims 2-3, 11-12, 15-16 and 21-26 are also allowable for at least those same reasons.

Regarding claims 4-9, 14 and 17-19, applicants assert that the references cannot be combined in the manner suggested. Furthermore, even if for argument sake such a combination were possible, the combination would still fail to teach many of the elements of the claims. Also, the 35 U.S.C. 103(a) rejection depends from the above stated 35 U.S.C. 102(b) rejection. Insofar as the applicants have traversed the 35 U.S.C. 102(b) rejection, the 35 U.S.C. 103(a) rejection should be withdrawn.

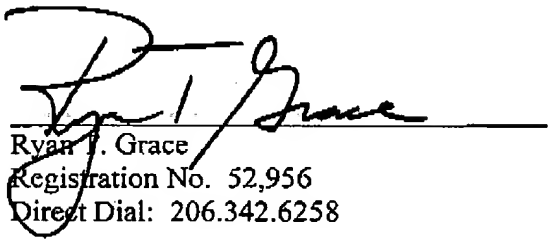
II. Request For Reconsideration

In view of the foregoing, all pending claims are believed to be allowable and the application is in condition for allowance. Therefore, further reconsideration and a Notice of Allowance is respectfully requested. Should the Examiner have any further issues regarding this

application, the Examiner is requested to contact the undersigned attorney for the applicants at the telephone number provided below.

Respectfully submitted,

MERCHANT & GOULD P.C.



Ryan T. Grace
Registration No. 52,956
Direct Dial: 206.342.6258

MERCHANT & GOULD P.C.
P. O. Box 2903
Minneapolis, Minnesota 55402-0903
206.342.6200